Lending Club Case Study

By Pradeep Kumar

Lending Club Dataset:

- This dataset has dimension as (39717, 111) including the current loan status (**Current, Charged-off, Fully Paid).** Loan Characteristics such as **loan amount, term, purpose** will help us in finding loan default. Borrower profile like age, relationship status and employment status are not relevant. Also, some of the features which are added in to the dataset after taking loans are very less significant like EMI, Delinquency, next payment date.

EDA on Lending Club Case

Here is the overview of the steps we are planning on EDA:

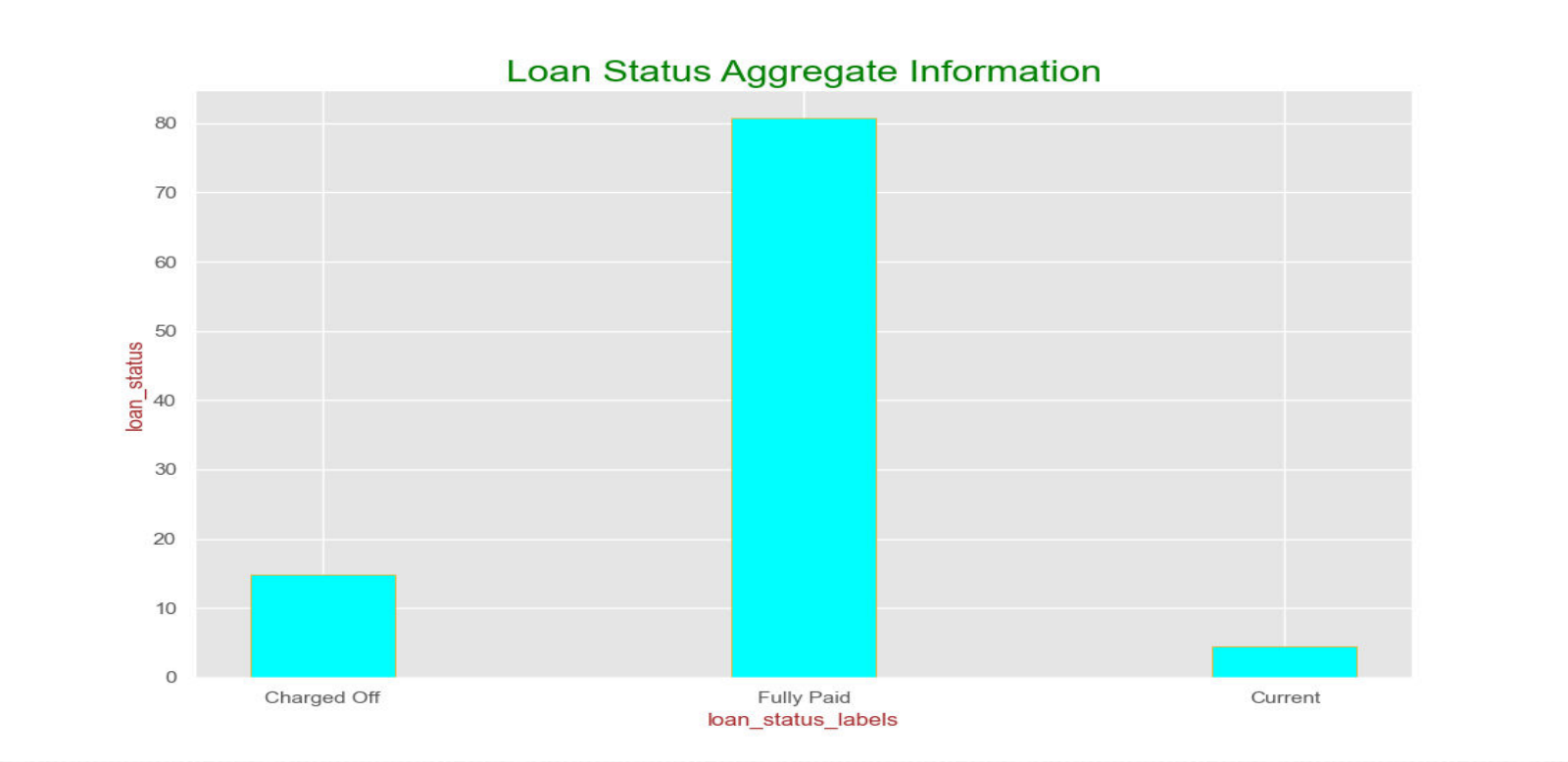
- **Dataset Overview**
- **Data Cleaning**
- **Metrics Derivation**
- **Univariate Analysis**
- **Bivariate Analysis**
- **Multivariate Analysis**

Data Sourcing

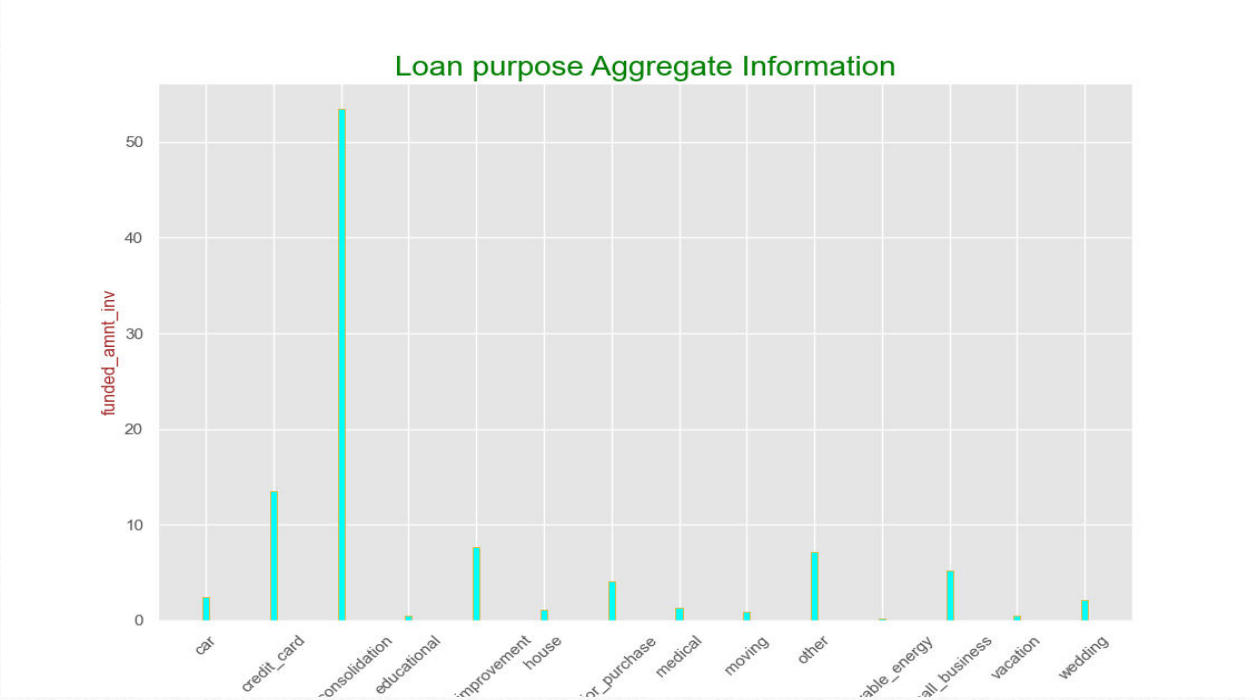
```
# Reading the DataSet
case_data = "loan.csv"
loan = pd.read_csv(case_data, low_memory=False)
print(loan.shape)
```

- **This dataset has dimension as (39717, 111)**

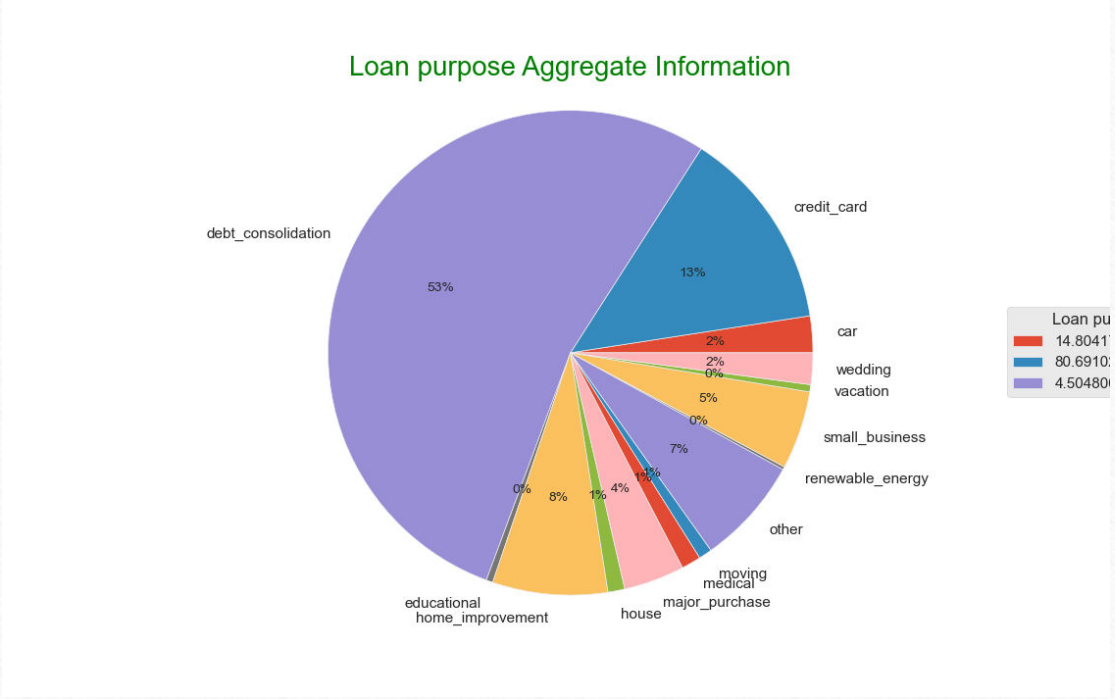
Data Summary(Loan Status Aggregation)



Data Summary(Loan purpose Aggregation)



Data Summary



Data Cleaning

Remove all columns having no values and remove all the columns which is not required

```
# DATA CLEANING

# remove all columns having no values

 # Remove the columns not required

  loan = loan.dropna(axis=1, how="all")

columnsnot_required = ["id", "member_id", "funded_amnt", "emp_title",
"pymnt_plan", "url", "desc", "title", "zip_code", "delinq_2yrs", "mths_since_last_delinq",
"mths_since_last_record", "revol_bal", "initial_list_status", "out_prncp",
"out_prncp_inv", "total_pymnt", "total_pymnt_inv", "total_rec_prncp", "total_rec_int",
"total_rec_late_fee", "recoveries", "collection_recovery_fee", "last_pymnt_d",
"last_pymnt_amnt", "next_pymnt_d", "last_credit_pull_d",
"collections_12_mths_ex_med", "policy_code", "acc_now_delinq",
"chargeoff_within_12_mths", "delinq_amnt", "tax_liens", "application_type",
"pub_rec_bankruptcies", "addr_state"]

loan.drop(columnsnot_required, axis=1, inplace=True)
```

- **columns left for the analysis now:**

['loan_amnt', 'funded_amnt_inv', 'term', 'int_rate', 'installment', 'grade', 'sub_grade', 'emp_length', 'home_ownership', 'annual_inc', 'verification_status', 'issue_d', 'loan_status', 'purpose', 'dti', 'earliest_cr_line', 'inq_last_6mths', 'open_acc', 'pub_rec', 'revol_util', 'total_acc']

Data Cleaning

Now need to remove with null values and need to remove records with loan status as Current as it is not significant as it is running.

```
# remove the rows with null values
loan.dropna(axis=0, subset=["emp_length"], inplace=True)
# remove NA rows for revol_util
loan.dropna(axis=0, subset=["revol_util"], inplace=True)
# Need to remove the rows with loan_status as "Current"
loan = loan[loan["loan_status"].apply(lambda x:False if x == "Current" else True)]
```

Data Cleaning

Update the loan status as fully paid and Charged off as a 0 and 1. Also update employee length as 0 and 1.Need to check the purpose value count and need to remove the values on the particular purpose.

```
# update loan_status as Fully Paid to 0 and Charged Off to 1

loan["loan_status"] = loan["loan_status"].apply(lambda x: 0 if x == "Fully Paid" else 1)

# update emp_length feature with continuous values as int

# where less 1 year= 0 and 10+ =10

loan["emp_length"] = pd.to_numeric(loan["emp_length"].apply(lambda x:0 if "<" in x
else (x.split('+')[0] if "+" in x else x.split()[0])))

# look through the purpose value counts

loan_purpose_values = loan["purpose"].value_counts()*100/loan.shape[0]

# remove rows with less than 1% of value counts in paricular purpose

loan_purpose_delete = loan_purpose_values[loan_purpose_values<1].index.values

loan = loan[[False if p in loan_purpose_delete else True for p in loan["purpose"]]]
```

Metric Derivation

Issue date is not in the standard format also we can split the date into two columns with month and the year which will make it easy for analysis

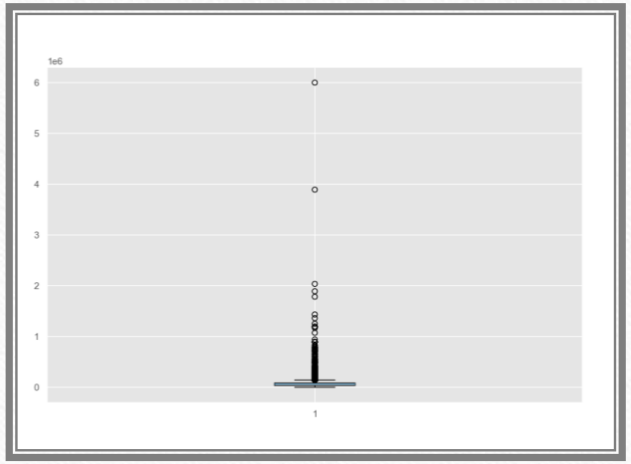
```
from datetime import datetime
def formatdate(datetime):
    year = datetime.split("-")[0]
    if(len(year) == 1):
        datetime = "0"+datetime
    return datetime
loan['issue_d'] = loan['issue_d'].apply(lambda x:formatdate(x))
loan['issue_d'] = loan['issue_d'].apply(lambda x: datetime.strptime(x, '%b-%y'))
# extracting month and year from issue_date
loan['month'] = loan['issue_d'].apply(lambda x: x.month)
print(loan['month'])
loan['year'] = loan['issue_d'].apply(lambda x: x.year)
print(loan['year'])


loan["earliest_cr_line"] = pd.to_numeric(loan["earliest_cr_line"].apply(lambda x:x.split('-')[1]))
print(loan["earliest_cr_line"])
```

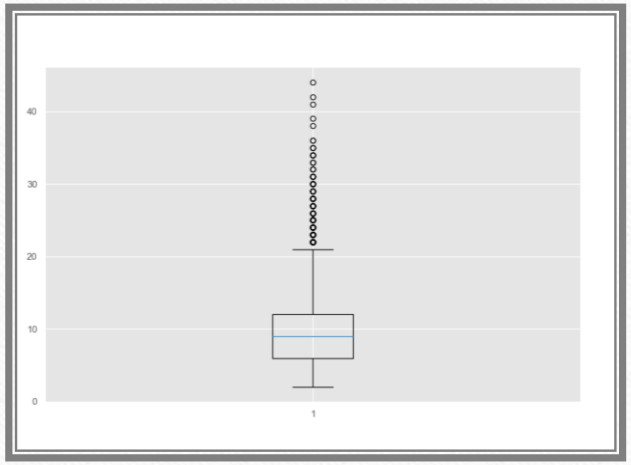
Outlier Treatment

To remove the outliers, we can use IQR method (Q1–1.5*IQR to Q3+1.5*IQR) we need to remove outliers from annual_inc i.e. 99 to 100%

1) For annual_inc features:

```
plt.boxplot(loan["annual_inc"])
plt.show()
annual_inc_q = loan["annual_inc"].quantile(0.99)
loan = loan[loan["annual_inc"] < annual_inc_q]
plt.boxplot(loan["annual_inc"])
plt.show()
```



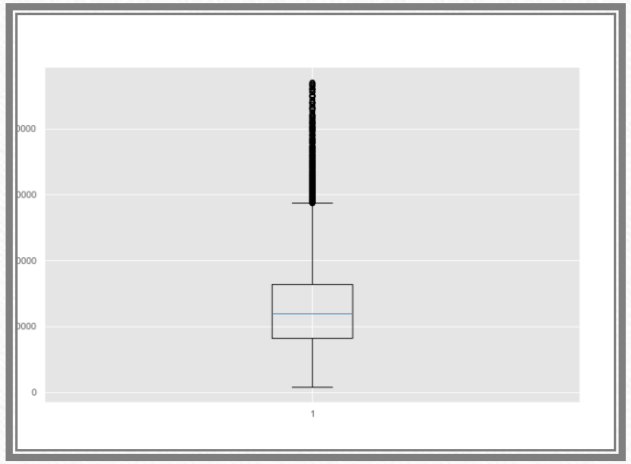


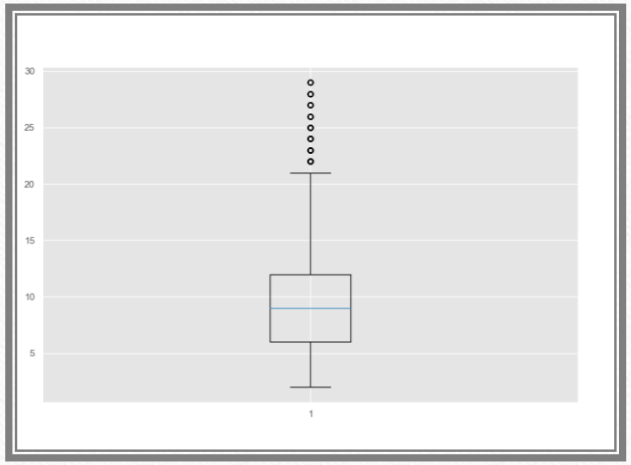
Outlier Treatment

To remove the outliers, we can use IQR method (Q1–1.5*IQR to Q3+1.5*IQR) we need to remove outliers from annual_inc i.e. 99 to 100%

2) For open_acc features:

```
plt.boxplot(loan["open_acc"])
plt.show()
open_acc_q = loan["open_acc"].quantile(1.0)
loan = loan[loan["open_acc"] < open_acc_q]
plt.boxplot(loan["open_acc"])
plt.show()
```



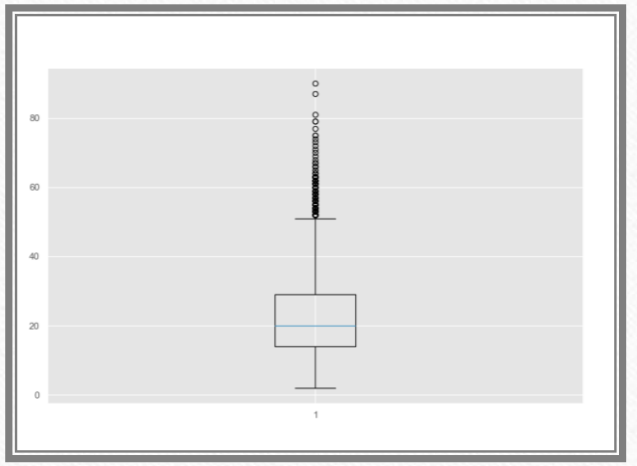


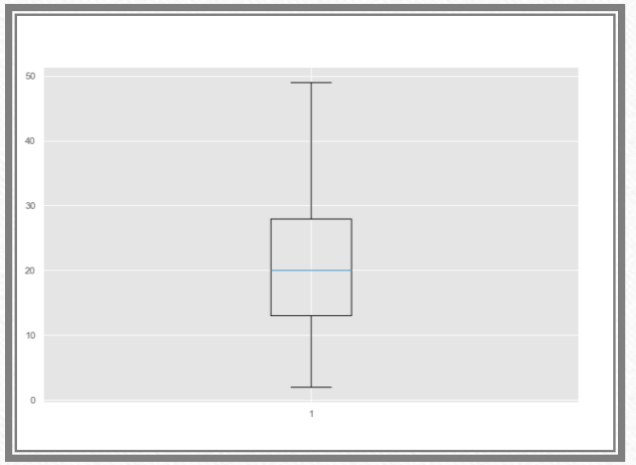
Outlier Treatment

To remove the outliers, we can use IQR method (Q1–1.5*IQR to Q3+1.5*IQR) we need to remove outliers from annual_inc i.e. 99 to 100%

2) For total_acc features:

```
plt.boxplot(loan["total_acc"])
plt.show()
total_acc_q = loan["total_acc"].quantile(0.98)
loan = loan[loan["total_acc"] < total_acc_q]
plt.boxplot(loan["total_acc"])
plt.show()
```



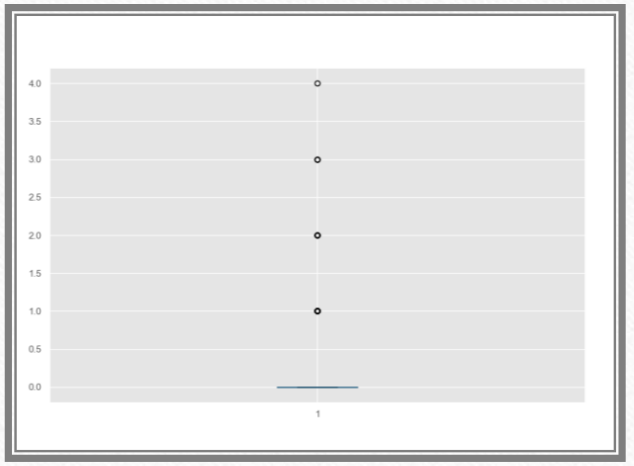


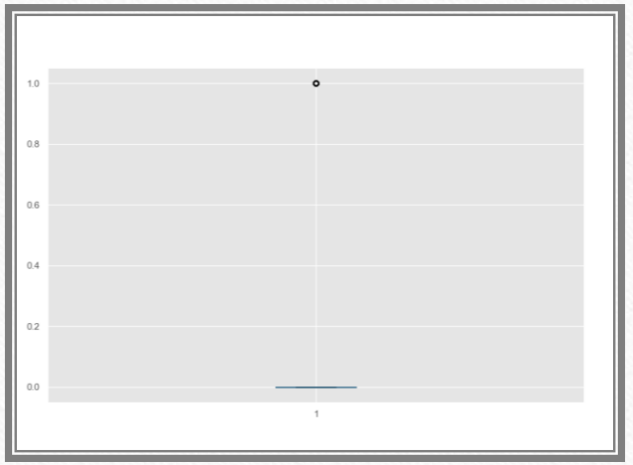
Outlier Treatment

To remove the outliers, we can use IQR method (Q1–1.5*IQR to Q3+1.5*IQR) we need to remove outliers from annual_inc i.e. 99 to 100%

3) For pub_rec features:

```
plt.boxplot(loan["pub_rec"])
plt.show()
pub_rec_q = loan["pub_rec"].quantile(0.995)
loan = loan[loan["pub_rec"] <= pub_rec_q]
plt.boxplot(loan["pub_rec"])
plt.show()
```





Deriving Metrics

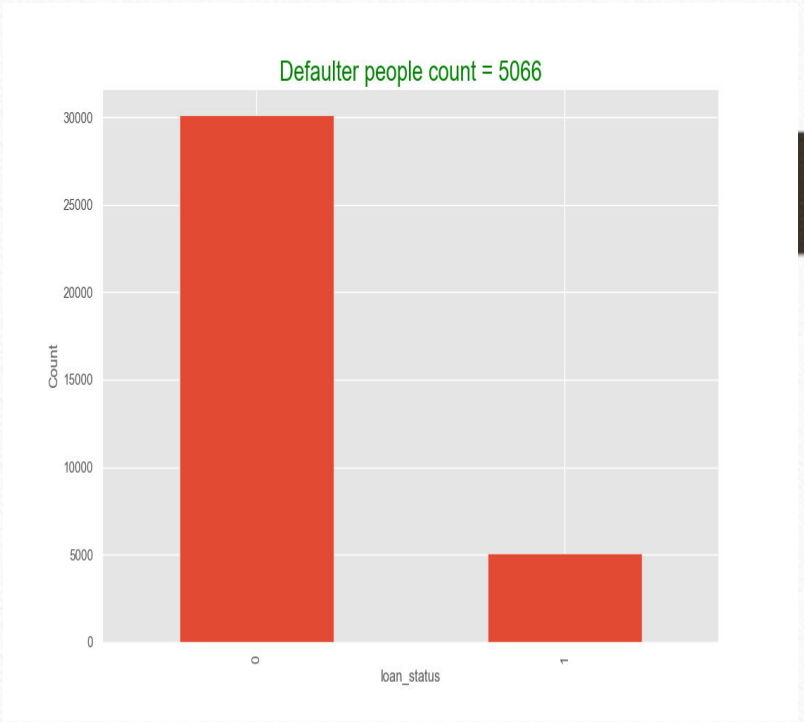
Data driven metrices as Date time formatting in a useful format and extracting year and month

```
#Date formatting
from datetime import datetime
def formatdate(datetime):
    year = datetime.split("-")[0]
    if(len(year) == 1):
        datetime = "0"+datetime
    return datetime
loan['issue_d'] = loan['issue_d'].apply(lambda x:formatdate(x))
loan['issue_d'] = loan['issue_d'].apply(lambda x: datetime.strptime(x, '%b-%y'))
# extracting month and year from issue_date
loan['month'] = loan['issue_d'].apply(lambda x: x.month)
print(loan['month'])
loan['year'] = loan['issue_d'].apply(lambda x: x.year)
print(loan['year'])
loan["earliest_cr_line"] = pd.to_numeric(loan["earliest_cr_line"].apply(lambda x:x.split('-')[1]))
print(loan["earliest_cr_line"])
```

Defaulters' data visualization through a Bar chart

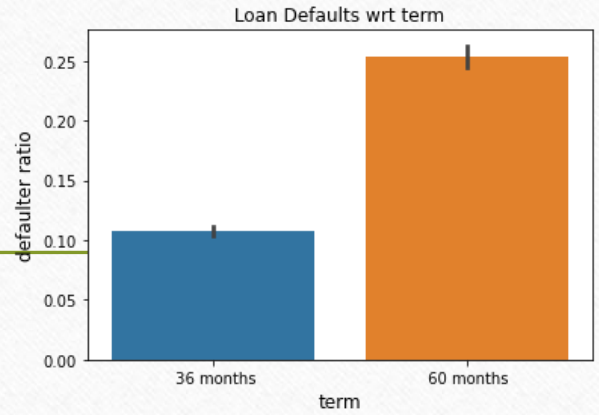
```
#  Check for amount of defaults in the data using barplot
defaulter=loan["loan_status"].value_counts()
loan["loan_status"].value_counts().plot.bar(xlabel="loan_status", ylabel="Count")
plt.title('Defaulter people count = '+str(defaulter[1]),fontdict={'fontsize':20,'fontweight':5,'color':'GREEN'})
plt.show()
```

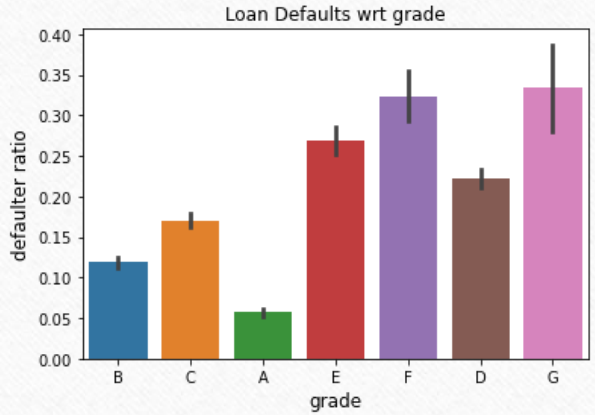
Conclusion: From the above bar chart we can conclude that 5066 people are defaulters which is near about ~14.5 % of the total records 35152.



Univariate Analysis

```
def univariateBarPlot(x):
    sns.barplot(x=x, y='loan_status', data=loan)
    plt.title("Loan Defaults wrt "+str(x))
    plt.xlabel(x, fontsize=12)
    plt.ylabel("defaulter ratio", fontsize=12)
# check for defaulters wrt term
univariateBarPlot("term")
plt.show()
# check for defaulters wrt grade
univariateBarPlot("grade")
plt.show()
```





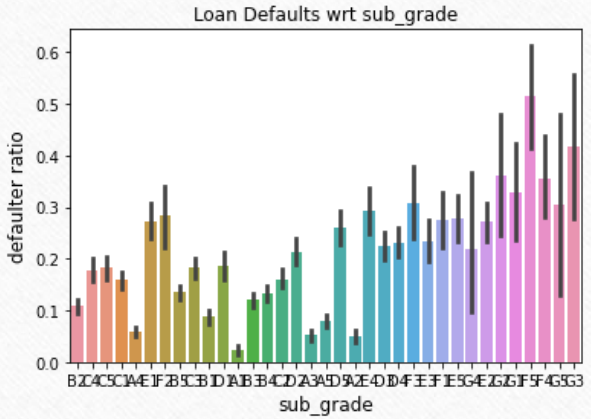
Conclusion:

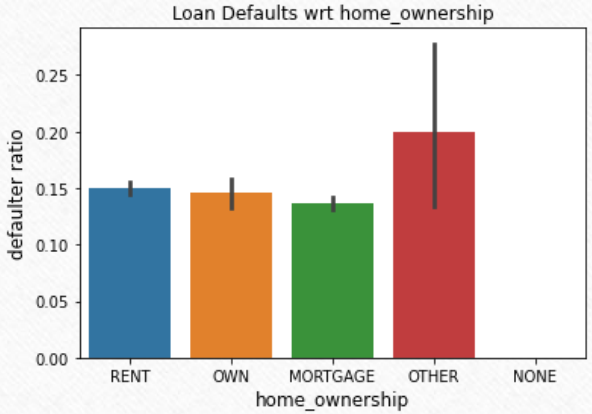
is term benificial -> Yes. *As defaulters rate is increasing wrt term, hence the chances of loan getting deaulted is less for 36m than 60m.*

is grade benificial -> Yes. As *defaulters rate is increasing wrt grade, hence the chances of loan getting deaulted increases with the grade.*

Univariate Analysis

```
def univariateBarPlot(x):
    sns.barplot(x=x, y='loan_status', data=loan)
    plt.title("Loan Defaults wrt "+str(x))
    plt.xlabel(x, fontsize=12)
    plt.ylabel("defaulter ratio", fontsize=12)
# check for defaulters wrt sub_grade
univariateBarPlot("sub_grade")
plt.show()
# check for defaulters wrt home_ownership
univariateBarPlot("home_ownership")
plt.show()
```





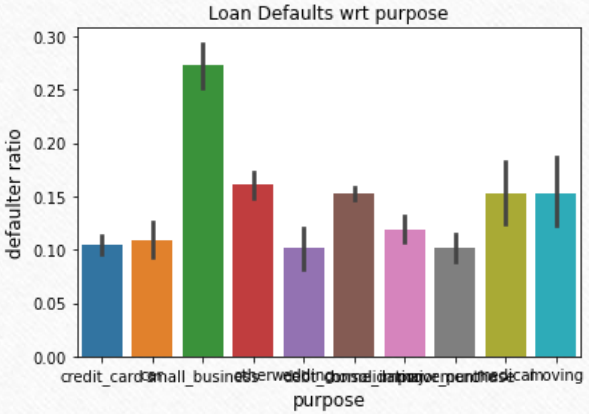
Conclusion:

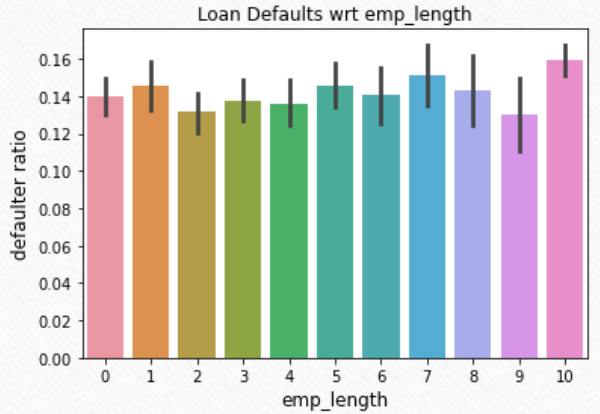
is sub_grade benificial -> Yes. *As defaulters rate is increasing wrt sub_grade, hence the chances of loan getting deaulted increases.*

is home_ownership benificial -> No. *As defaulters rate is constant here*

Univariate Analysis

```
def univariateBarPlot(x):
    sns.barplot(x=x, y='loan_status', data=loan)
    plt.title("Loan Defaults wrt "+str(x))
    plt.xlabel(x, fontsize=12)
    plt.ylabel("defaulter ratio", fontsize=12)
# check for defaulters wrt purpose
univariateBarPlot("purpose")
plt.show()
# check for defaulters wrt emp_length
univariateBarPlot("emp_length")
plt.show()
```





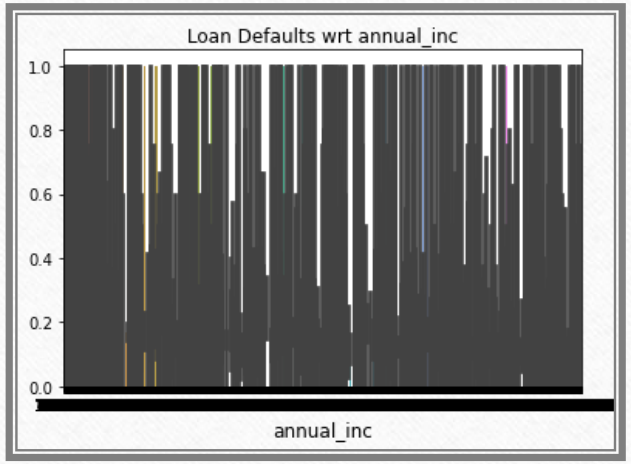
Conclusion:

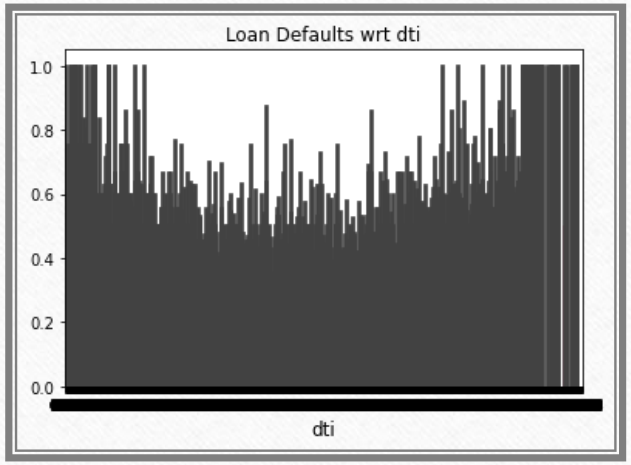
is purpose benificial -> Yes. As *the defaulters rate is nearly constant for all purpose type except 'small business', hence rate will depend on purpose of the loan*

is emp_length benificial -> No. As *the defaulters rate is constant here, hence defaulter does not depends on emp_length*

Univariate Analysis

```
def univariateBarPlot(x):
    sns.barplot(x=x, y='loan_status', data=loan)
    plt.title("Loan Defaults wrt "+str(x))
    plt.xlabel(x, fontsize=12)
    plt.ylabel("defaulter ratio", fontsize=12)
# check for defaulters wrt annual_inc
univariateBarPlot("annual_inc")
plt.show()
# check for defaulters wrt dti
univariateBarPlot("dti")
plt.show()
```





Conclusion:

is annual_inc_range benificial -> Yes. As the defaulters rate is decreasing as with annual_inc_range values

is dti_range benificial -> Yes. As the defaulters rate is increasing as with dti_range values

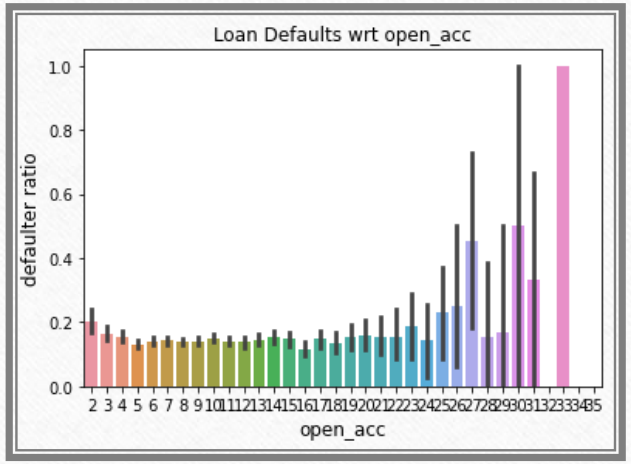
Univariate Analysis

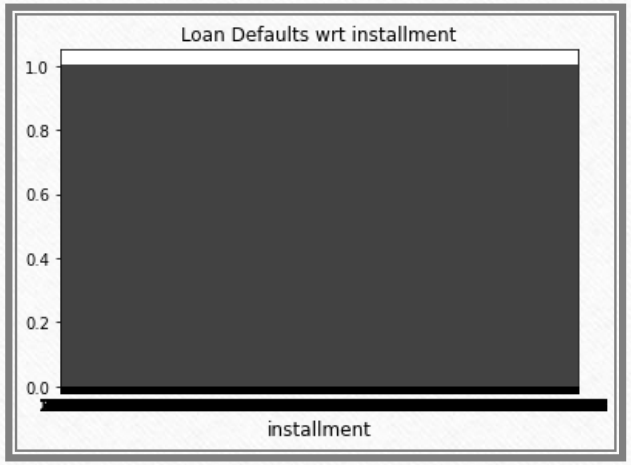
```
def univariateBarPlot(x):
    sns.barplot(x=x, y='loan_status', data=loan)
    plt.title("Loan Defaults wrt "+str(x))
    plt.xlabel(x, fontsize=12)
    plt.ylabel("defaulter ratio", fontsize=12)
# check for defaulters wrt open_acc
univariateBarPlot("open_acc")
plt.show()
# check for defaulters wrt installment
univariateBarPlot("installment")
plt.show()
```

Conclusion:

is open_acc benificial -> No. As the defaulters rate is nearly constant for feature open_acc

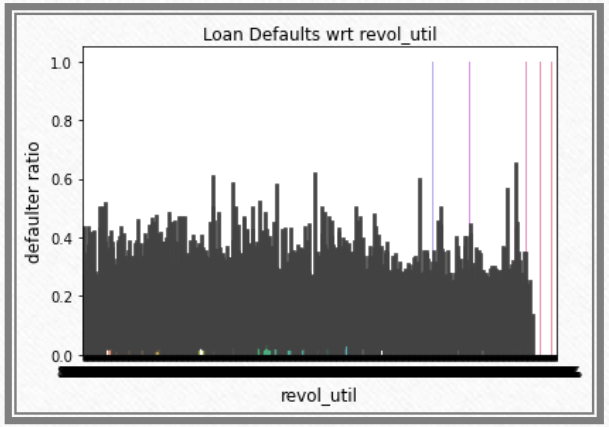
is installment benificial -> Yes. As the defaulters rate is increasing as with installment values





Univariate Analysis

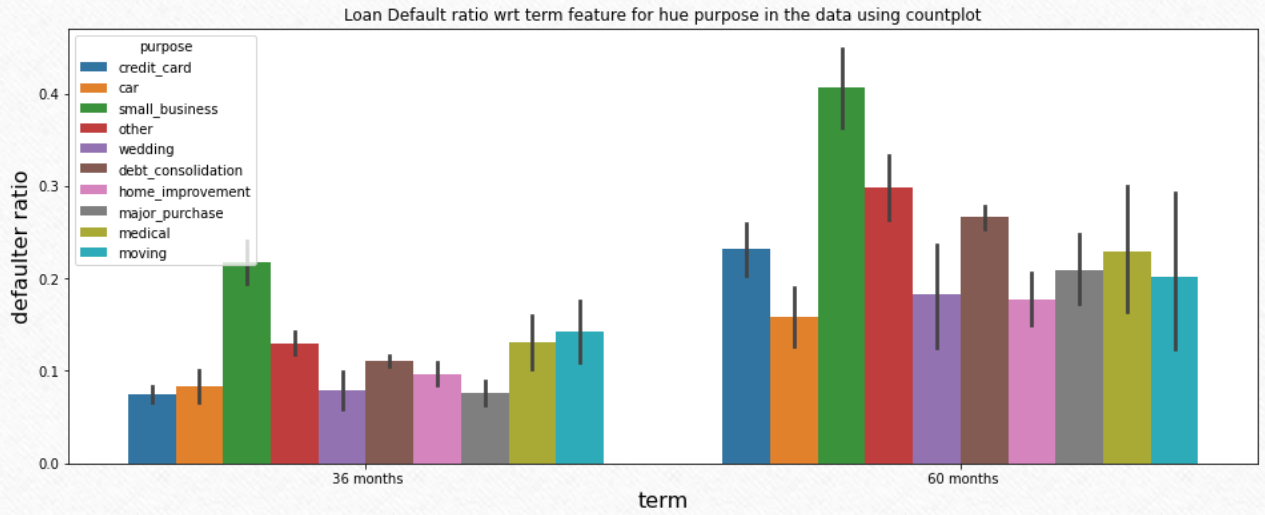
```
#univariate plots
def univariateBarPlot(x):
    sns.barplot(x=x, y='loan_status', data=loan)
    plt.title("Loan Defaults wrt "+str(x))
    plt.xlabel(x, fontsize=12)
    plt.ylabel("defaulter ratio", fontsize=12)
# check for defaulters wrt revol_util
univariateBarPlot("revol_util")
plt.show()
```



Conclusion: ***is revol_util benificial -> Yes. As the defaulters rate is fluctuating where some have complete 100% ratio for defaulter and is increasing as the magnitude increases***

Bivariate Analysis

```
# function to plot scatter plot for two features
def plotScatter(x, y):
    plt.figure(figsize=(16,6))
    sns.scatterplot(x=x, y=y, hue="loan_status", data=loan)
    plt.title("Scatter plot between "+x+" and "+y)
    plt.xlabel(x, fontsize=16)
    plt.ylabel(y, fontsize=16)
    plt.show()
def plotBivariateBar(x, hue, figsize=(16,6)):
    plt.figure(figsize=figsize)
    sns.barplot(x=x, y='loan_status', hue=hue, data=loan)
    plt.title("Loan Default ratio wrt "+x+" feature for hue "+hue+" in the data using countplot")
    plt.xlabel(x, fontsize=16)
    plt.ylabel("defaulter ratio", fontsize=16)
    plt.show()
```



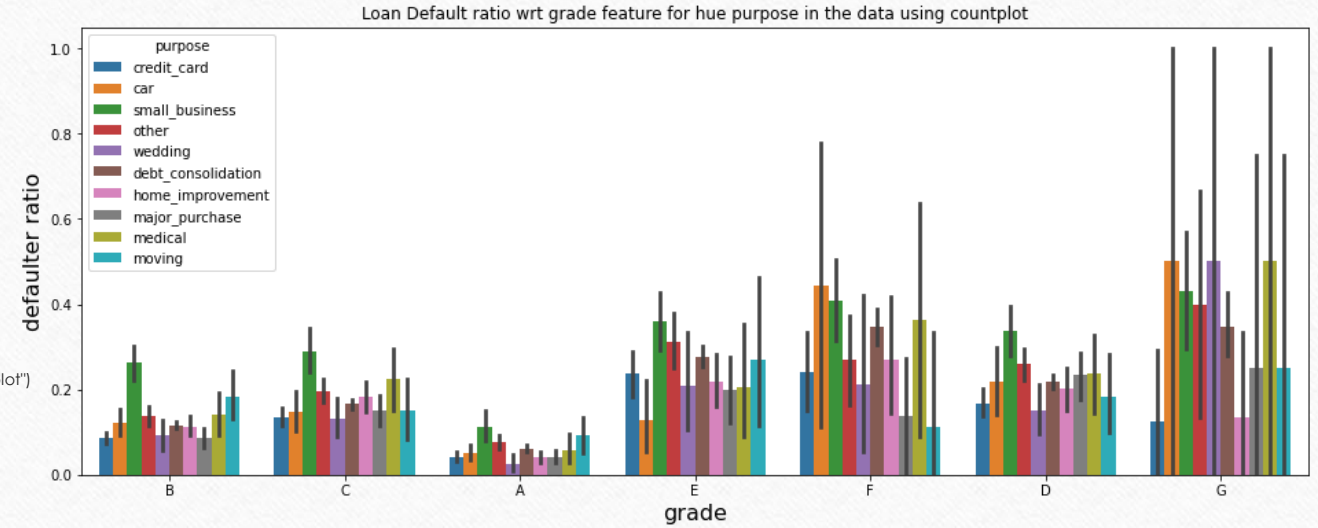
 # check for defaulters wrt term and purpose in the data

plotBivariateBar("term", "purpose")

Conclusion: is *related – Yes. As default ratio increases for every purpose wrt term*

Bivariate Analysis

```
# function to plot scatter plot for two features
def plotScatter(x, y):
    plt.figure(figsize=(16,6))
    sns.scatterplot(x=x, y=y, hue="loan_status", data=loan)
    plt.title("Scatter plot between "+x+" and "+y)
    plt.xlabel(x, fontsize=16)
    plt.ylabel(y, fontsize=16)
    plt.show()
def plotBivariateBar(x, hue, figsize=(16,6)):
    plt.figure(figsize=figsize)
    sns.barplot(x=x, y='loan_status', hue=hue, data=loan)
    plt.title("Loan Default ratio wrt "+x+" feature for hue "+hue+" in the data using countplot")
    plt.xlabel(x, fontsize=16)
    plt.ylabel("defaulter ratio", fontsize=16)
    plt.show()
```



 # check for defaulters wrt grade and purpose in the data

plotBivariateBar("grade", "purpose")

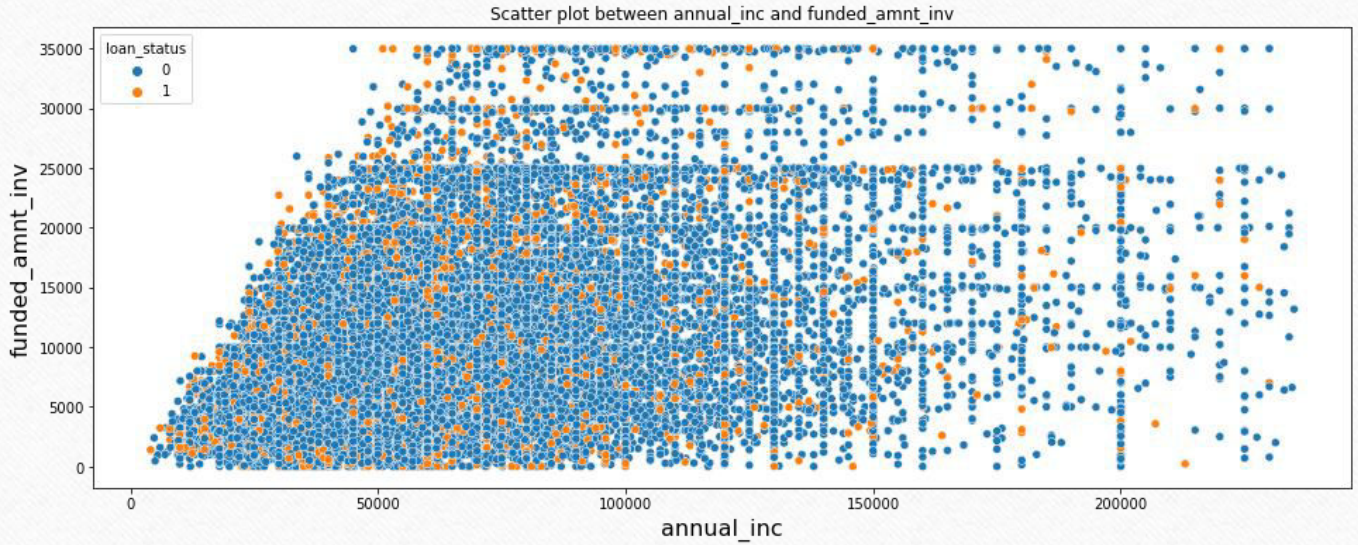
Conclusion: is *related – Yes. As default ratio increases for every purpose wrt grade*

Bivariate Analysis

```
# function to plot scatter plot for two features
def plotScatter(x, y):
    plt.figure(figsize=(16,6))
    sns.scatterplot(x=x, y=y, hue="loan_status", data=loan)
    plt.title("Scatter plot between "+x+" and "+y)
    plt.xlabel(x, fontsize=16)
    plt.ylabel(y, fontsize=16)
    plt.show()
def plotBivariateBar(x, hue, figsize=(16,6)):
    plt.figure(figsize=figsize)
    sns.barplot(x=x, y='loan_status', hue=hue, data=loan)
    plt.title("Loan Default ratio wrt "+x+" feature for hue "+hue+" in the data using countplot")
    plt.xlabel(x, fontsize=16)
    plt.ylabel("defaulter ratio", fontsize=16)
    plt.show()
```

 # plot scatter for funded_amnt_inv with annual_inc

plotScatter("annual_inc", "funded_amnt_inv")



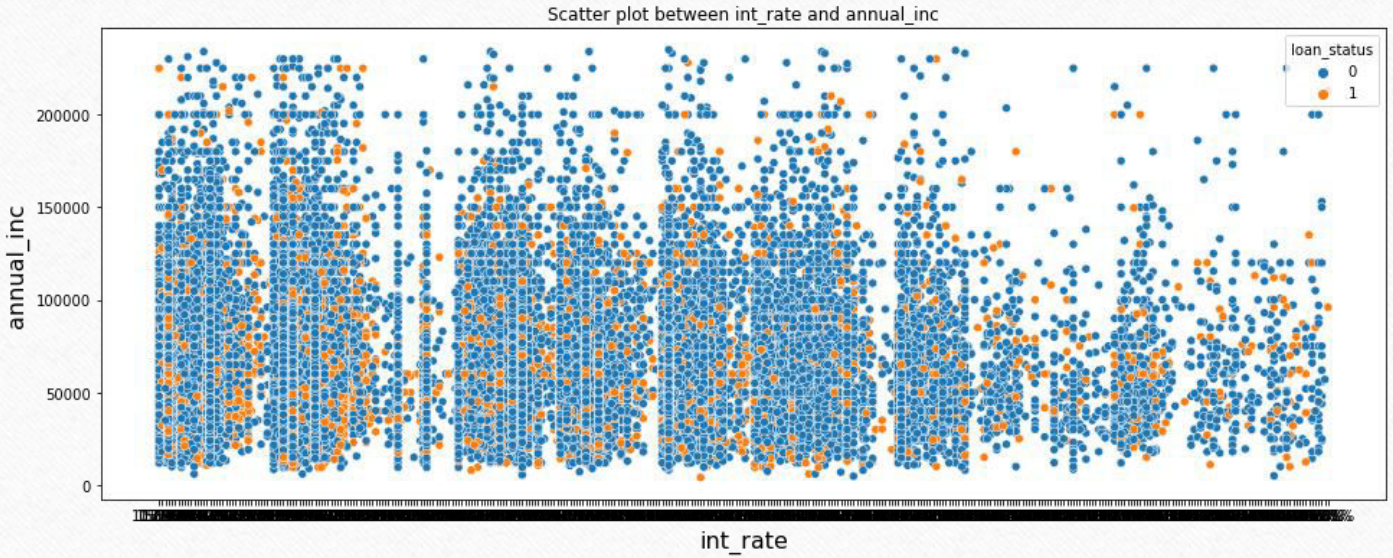
Conclusion: is *related – Yes. As default ratio increases for every purpose wrt annual_inc_range*

Bivariate Analysis

```
# function to plot scatter plot for two features
def plotScatter(x, y):
    plt.figure(figsize=(16,6))
    sns.scatterplot(x=x, y=y, hue="loan_status", data=loan)
    plt.title("Scatter plot between "+x+" and "+y)
    plt.xlabel(x, fontsize=16)
    plt.ylabel(y, fontsize=16)
    plt.show()
def plotBivariateBar(x, hue, figsize=(16,6)):
    plt.figure(figsize=figsize)
    sns.barplot(x=x, y='loan_status', hue=hue, data=loan)
    plt.title("Loan Default ratio wrt "+x+" feature for hue "+hue+" in the data using countplot")
    plt.xlabel(x, fontsize=16)
    plt.ylabel("defaulter ratio", fontsize=16)
    plt.show()
```

 # plot scatter for int_rate with annual_inc

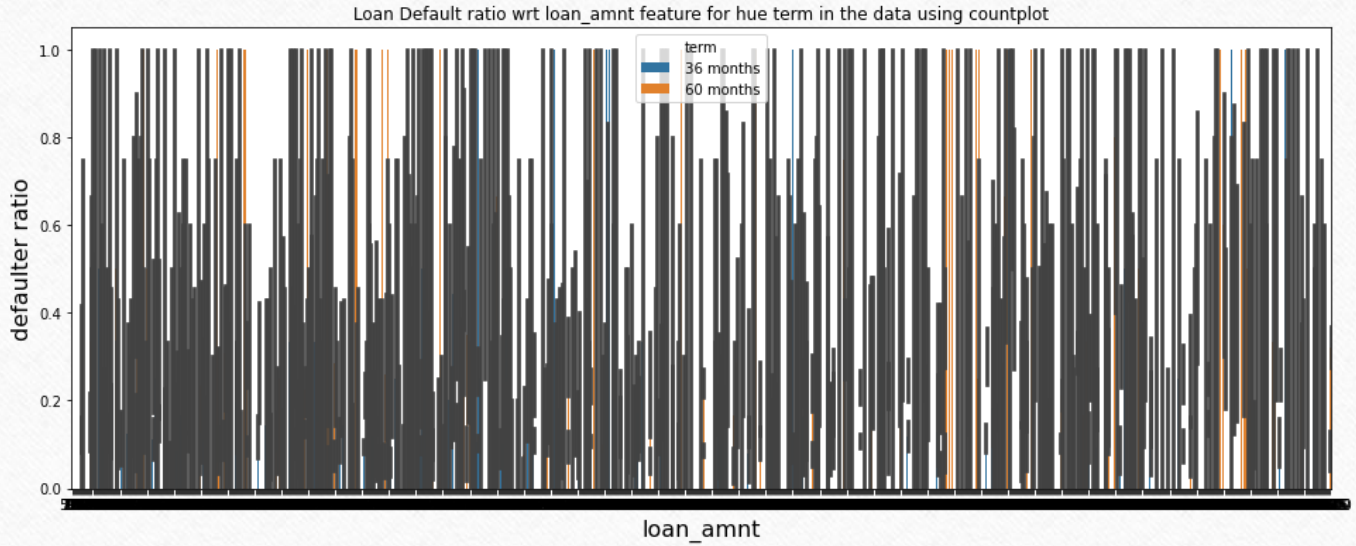
plotScatter("int_rate", "annual_inc")



Conclusion: is *related* – No. As *there is no relation between above mentioned features*

Bivariate Analysis

```
# function to plot scatter plot for two features
def plotScatter(x, y):
    plt.figure(figsize=(16,6))
    sns.scatterplot(x=x, y=y, hue="loan_status", data=loan)
    plt.title("Scatter plot between "+x+" and "+y)
    plt.xlabel(x, fontsize=16)
    plt.ylabel(y, fontsize=16)
    plt.show()
def plotBivariateBar(x, hue, figsize=(16,6)):
    plt.figure(figsize=figsize)
    sns.barplot(x=x, y='loan_status', hue=hue, data=loan)
    plt.title("Loan Default ratio wrt "+x+" feature for hue "+hue+" in the data using countplot")
    plt.xlabel(x, fontsize=16)
    plt.ylabel("defaulter ratio", fontsize=16)
    plt.show()
```



 # check for defaulters wrt loan_amnt_range and term in the data

plotBivariateBar("loan_amnt", "term")

Conclusion: Is related->Yes. As *default ratio increases for every term wrt loan_amnt_range*

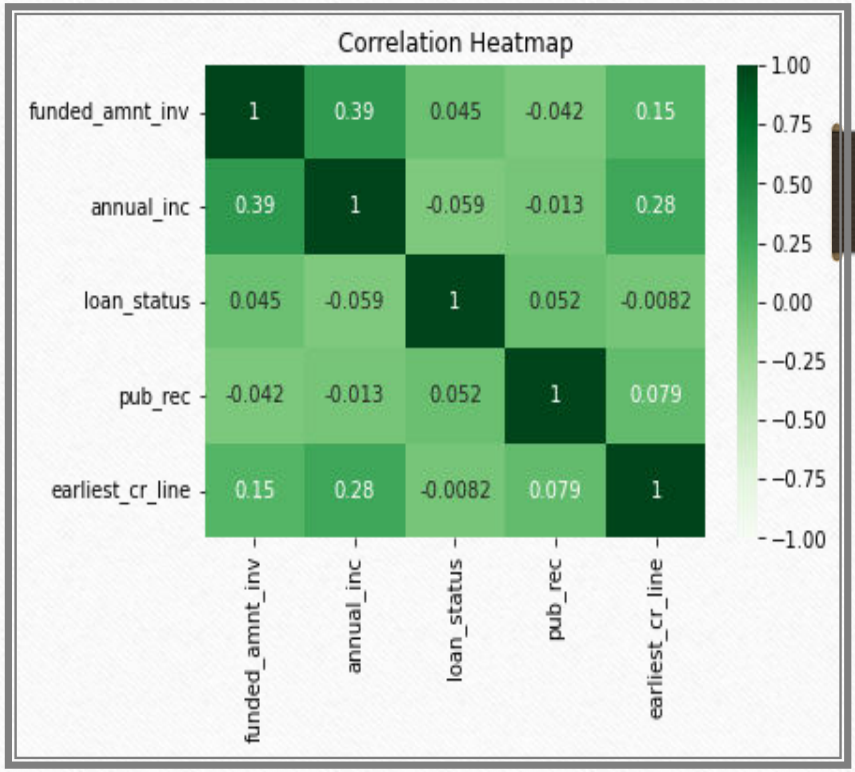
Multivariate Analysis(Correlation Heatmap)

 # plot heat map to see correlation between features

```
continuous_f = ["funded_amnt_inv", "annual_inc", "term", "int_rate", "loan_status", "revol_util", "pub_rec", "earliest_cr_line"]

loan_corr = loan[continuous_f].corr()

sns.heatmap(loan_corr,vmin=-1.0,vmax=1.0,annot=True, cmap = "Greens")

plt.title("Correlation Heatmap")

plt.show()
```

Conclusion: Important related feature from above Multivariate analysis are:

[term, grade, purpose, revol_util, int_rate, installment, annual_inc, funded_amnt_inv]



Conclusion

After all the Exploratory data analysis on the features available in the dataset, we have have arrived to the conclusion that *the best driving features for the Loan default analysis are below:*

1) term,
2) grade,
3) purpose,
4) revol_util, int_rate,
5) installment,
6) annual_inc and
7) funded_amnt_inv


THANK
YOU!